University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

Object Oriented Programming/ Lab
البرمجة الشيئية/ مختبر

Assist. Prof. Dr. Ekhlas Falih Naser
أ.م.د. أخلاص فالح ناصر

cs.uotechnology.edu.iq

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih — Branch: NW and CS

Subject: OOP — Class: Second

Week no.: 1 — Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Function, Function overloading and default arguments

A function is a set of statements designed to accomplish a particular task. Overloading refers to the use the same thing for different purposes. The function declaration must provide default values for those arguments that are not specified.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***Function, Function overloading and default arguments*** using C++ language.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

Example 1: Write a C++ program to find the maximum value between two values using function

```
#include <iostream.h>
float max ()
{
   float  a,  b;        a=3.5;        b=10.25;
    float  c;
    if (a > b)
          c = a;
      else
          c = b;
    return (c);
}
void main (  )
{    float k;
       k=max();
     cout <<k;
}
```

Output:-
10.25

Example2:Write a c++program to find the volumes using function overloading

```
#include <iostream.h>

int       volume(int);
double volume(double,int);
long     volume(long,int,int);

void main( )
{
        cout<<volume(10)<<"\n";
        cout<<volume(2.5,8)<<"\n";
        cout<<volume(100L,75,15);
}

int volume(int s)
{
        return(s*s*s);                    // cube
}

double volume(double r,int h)        // cylinder
{
        return(3.14519*r*r*h);
}

long volume(long l,int b,int h)       // rectangular box
{
        return(l*b*h);
}
```

Example 3:Write a simple program to represent a default argument

```
#include <iostream>
using namespace std;

void repchar(char='*', int=45);    //declaration with
                                   //default arguments
int main()
   {
   repchar();                      //prints 45 asterisks
   repchar('=');                   //prints 45 equal signs
   repchar('+', 30);               //prints 30 plus signs
   return 0;
   }
//--------------------------------------------------------------
// repchar()
// displays line of characters
void repchar(char ch, int n)       //defaults supplied
   {                               //    if necessary
   for(int j=0; j<n; j++)          //loops n times
      cout << ch;                  //prints ch
   cout << endl;
   }
```

توقيع تدريسي المادة: توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih Branch: NW and CS

Subject: OOP Class: Second

Week no.: 2 Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Class definition, constructor and Destructor

Classes are collections of variables and functions that operate on those variables. The variables in a class definition are called data members, and the functions are called member functions.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***Class*** function using C++ language and ***constructor and Destructor***.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

Example1: write an OOP to define the coordinate of point and shift these values of this point.

```
#include<iostream.h>

class point {
      int xval,yval;
public:
     void setpt(int x,int y)
{
     xval=x;
     yval=y;
}
     void offsetpt(int x,int y)
{
     xval+=x;

     yval+=y;

          cout<<xval<<yval;

}
};
void main()
{
   point pt;

 pt.setpt(10,20);

 pt.offsetpt(2,2);
}
```

Example 2: Write an OOP to find the area of rectangle using constructor and the destructor.

```
# include <iostream.h>

class Rectangle
{
   int length , width;
   public:
    Rectangle(int x, int y)              //constructor
{
   length = x;
   width = y;
}
    ~ Rectangle()                       //destructor
    {

    }


int area( )
{
   return (length *width);
}

};


void main( )
{
   Rectangle rect1(6,7);
   cout<< rect1.area( ) << endl;
}
```

توقيع تدريسي المادة: توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih — Branch: NW and CS

Subject: OOP — Class: Second

Week no.: 3 — Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Friend function

*Occasionally we may need to grant a function access to the nonpublic members of a class. Such an access is obtained by declaring the function a **friend** of the class.*

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement **Friend function** to one class or two classes using C++ language.

Example1: Write an OOP to find the summation of point coordinates using friend function class.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>

class point
{
private:
int xval,yval;
public:

point(int x,int y)
    {
        xval=x+2;
        yval=y+3;
    cout<<"xval= "<<xval<<" "<<"yval= "<<yval<<endl;

    }

 friend int sum(point p);                    //friend function

};

int sum(point p)
{
int s = p.xval + p.yval;
return (s);
}

void main( )
{
    point p(2,2);

    cout<<" the summation of coordinate x & y = "<<sum(p)<<endl;
}
```

Example 2: Write an OOP to print the coordinates of a class point and a class D3 using friend function.

```
#include <iostream.h>

class D3;

class point
{
private:
int xval,yval;
public:
 void get()
 {
     xval =6;          yval=3;
 }

    friend void write(point p, D3 d);       //friend function
};

class D3
{
private:
int zval;
public:
D3()
{  zval=5;

}
friend void write(point p, D3 d);         //friend function
};


 void write(point p, D3 d)          //function definition
{
  cout<<"xval= "<<p.xval<<endl;
  cout<<"yval= "<<p.yval<<endl;
  cout<<"zval= "<<d.zval<<endl;
}

void  main()
{
point p;

D3  d;

p.get();

write(p,d);

}
```

Output:

xval=6
yval=3
zval=5

توقيع تدريسي المادة: توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP | Class: Second

Week no.: 4 | Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Friend Class

*The member functions of a class can all be made friends at the same time when you make the entire **class a friend**.*

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement **Friend class** using C++ language.

Example 1:Write an OOP to divide the value m in a class first on the value n in the class second using a friend class.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include<iostream.h>
class first
{
     int m;
public:
     void get_m()
     {
          m=12;
     }
     friend class second;
};

class second
{
     int n;
public:
     void get_n()
     {
          n=6;
     }

     void div(first f)
     {
          int b= f.m /n;
          cout<<"division= "<<b<<endl;
     }
};
void main()
{
     first    f;        f.get_m();
     second   s;       s.get_n();

     s.div(f);
}
```

Example 2: Write an OOP to read the values x and y in a class read and write the values x and y in a class write using friend class

```
#include<iostream.h>

class read
{
    int x,y;
public:
     read()
    {
        cout<<"Enter the values of x & y";
            cin>>x>>y;
    }

    friend class write;
};

class write
{
public:

    void write_xy(read r)
    {

        cout<<r.x <<" "<<r.y<<endl;
    }
};
void main()
{
    read   r;

    write  w;

    w.write_xy(r);
}
```

توقيع تدريسي المادة: **توقيع مشرف المختبر:**

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP | Class: Second

Week no.: 5 | Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Scope Operator *Resolution and Static Members*

When calling a member function, we usually use an abbreviated syntax. For example: ***pt.OffsetPt(2,2); // abbreviated form This is equivalent to the full form: pt.Point::OffsetPt(2,2); // full form***

A data member of a class can be defined to be static. This ensures that there will be exactly one copy of the member, shared by all objects of the class.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement **Scope Operator *Resolution and Static Members*** using C++ language.

Example 1: write an OOP to find the area of rectangle using the concept of Scope Operator Resolution.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
# include <iostream.h>

class Rectangle
{
     int length , width;
     public:

          Rectangle();
          void area();
};

Rectangle :: Rectangle()
{
           cout<<"Enter Length and Width";
           cin>>length;
           cin>>width;
}

void Rectangle::area()
{
   cout<<length*width;
}

void main()
{
      Rectangle rect;
       rect.area();
}
```

Example 1: Write an OOP to represent a static members for count class

```
#include<iostream.h>
class count
{
private:
static int c1;
 int c2;
public:
count()
{
c2=0;
}

static void write_c1() //static function
{
cout <<"c1= "<<++c1<<endl;
}

void write_c2() //non-static function
{
cout <<"c2= " <<++c2 << endl;
}
};
//-----------------------------------------------------------

int count::c1 = 0;

////////////////////////////////////////////////////////////////
void main()
{
count n1;
n1.write_c1();
n1.write_c2();

count n2;
n2.write_c1();
n2.write_c2();

count n3;
n3.write_c1();
n3.write_c2();

}
```

Output:

c1=1
c2=1
c1=2
c2=1
c1=3
c2=1

توقيع تدريسي المادة:

توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih — Branch: NW and CS

Subject: OOP — Class: Second

Week no.: 6 — Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Objects Pointers and Class Object Member . *It has already been stated that a pointer is a variable which holds the memory address of another variable of any basic data type. It has been shown that how a pointer variable can be declared with a class. A data member of a class may be of a user-defined type, that is, an object of another class.*

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***Objects Pointers and Class Object Member*** using C++ language.

Example 1:Write an OOP to read and display student information using pointer.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class student
{
private:
     int stageno;
     int age;
     char sex;
     float height;
     float weight;
public:
     void getinfo();
     void disinfo();
};     //end of class definition
void student::getinfo()
{
     cout<<" Stage number :";     cin>>stageno;     cout<<endl;
     cout<<" Age:";               cin>>age;         cout<<endl;
     cout<< "Sex :";              cin>>sex;         cout<<endl;
     cout<<"Height :";            cin>>height;      cout<<endl;
     cout<<"Weight :";            cin>>weight;
}


void student::disinfo()
{
     cout<<"Stage number = ";     cout<<stageno;       cout<<"\n";
     cout<<" Age= ";              cout<<age;           cout<<"\n";
     cout<< "Sex = ";             cout<<sex;           cout<<"\n";
     cout<<"Height =";            cout<<height;        cout<<"\n";
     cout<<"Weight =";            cout<<weight;
}

void main()
{
student *ptr;
ptr=new student;
cout<<" enter the following information"<<endl;
ptr->getinfo();
cout<<" \n contents of class "<<endl;
ptr->disinfo();
}
```

Example 1:Write an OOP to find the area of square using class object member

```
# include <iostream.h>
class area
{
     int k;
public:
   area(int x)
 {
    k=x;

    cout<<"k= "<<k<<endl;
    cout<<"area of square=  "<<k*k<<endl;
    }
};
class square
{
private:
    area length;
public:

square(int x):length(x)
    {

    }

};

void main()
{   square  s(7);

}
```

Output:
area of square= 49

توقيع تدريسي المادة: **توقيع مشرف المختبر:**

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP | Class: Second

Week no.: 7 | Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Objects arrays and an array of Pointers to Objects

In C++, the definition of an array specifies a variable type and a name. But it includes another feature: a size. The size specifies how many data items the array will contain.A common programming construction is an array of pointers to objects. This arrangement allows easy access to a group of objects, and is more flexible than placing the objects themselves in an array.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***Objects arrays and an array of Pointers to Objects*** using C++ language.

Example 1: Write an OOP to read and write student information for 10 students objects.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
# include <iostream.h>

class student
{
    char  name[20];
    int age ;
    float average;
public:
 void get_data()
 {
     cin>>name;
     cin>>age;
     cin>>average;
 }
void print_data()
 {
     cout<<name <<" " <<age<<" "<<average<<endl;
 }

};
void main( )
{
    student   s[10];

    for (int i=0;i<10;i++)
    {
    s[i].get_data();
    s[i].print_data();
    }
}
```

Example 1: Write an OOP to read and write student information for 10 students using array of pointer to objects

```
# include <iostream.h>
class student
{
    char  name[20];
    int age ;
    float average;
public:
 void get_data()
 {
     cin>>name;
     cin>>age;
     cin>>average;
 }
void print_data()
 {
     cout<<name <<" " <<age<<" "<<average<<endl;
 }
};
void main( )
{
    student   *s[10];

    for (int i=0;i<10;i++)
    {
        s[i]=new student;

        (*s[i]).get_data();

        (*s[i]).print_data();
    }
}
```

توقيع تدريسي المادة: توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih Branch: NW and CS

Subject: OOP Class: Second

Week no.: 8 Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Operators Overloading _ Unary Operator

Unary operators act on only one operand. Examples of unary operators are the increment and decrement operators ++ and --, and the unary minus, as in -33..

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***Unary Operator*** function and ***Unary Operator with Return Object*** using C++ language.

Example 1: وصف البرنامج بصيغة مثال صريح

Write an OOP to increment the count variable in class counter using unary operator overloading (+ +).

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class Counter
{
private:
  int count;
public:
Counter()        // Constructor
{ count=5;
}
void operator ++()     //increment (prefix)
{
++count;
}
void write()
{
     cout<<count;
}
};
////////////////////////////////////////////////////////////
void main()
{
Counter c1, c2;
cout<<"\nc1=";
c1.write();
cout<<"\nc2=";
c2.write();
++c1;          //increment c1
++c2;          //increment c2
++c2;          //increment c2
cout<<"\nc1=";
c1.write();
cout<<"\nc2=";
c2.write();
}
```

Example 2:

وصف البرنامج بصيغة مثال صريح

Write an OOP to decrement the count variable in class counter using unary operator overloading (- -) and return the object to other object in main function using Nameless Temporary Object.

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class decrement
{
private:
int x;
public:

decrement (int y)        // Constructor
{ x=y;
}
decrement operator --()
{
--x;

return decrement(x);
}
void write()
{
     cout<<x<<endl;
}
};
//////////////////////////////////////////////////////////////
void main()
{
decrement m(7), n(0);

n = --m;

m.write();
n.write();
}
```

توقيع تدريسي المادة :

توقيع مشرف المختبر:

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Binary Operators Overloading _ Arithmetic and Comparison Binary Operator

a) Arithmetic Operators

Arithmetic operators consists of (+,-,, /)*

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement **** Operator*** function using C++ language to multiply two numbers. Also Learn how to build and implement ***+ Operator*** function using C++ language to add two areas of **Rectangle**

Example 1: وصف البرنامج بصيغة مثال صريح

Write an OOP to multiply two numbers using operator overloading (*)

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

Write an OOP to multiply two numbers using operator overloading *

```
class mult
{    int  x;
Public:
  mult (int k)
     { x=k; }
       mult operator * (mult m)
       {
            int b=x*m.x;
   return mult(b) ;
}
void print()
{ cout<<x; }
};
void  main()
{   mult a1(3), a2(5) , a3(0);
                                  a3=a1*a2;
a3.print();
}
```

Example 2:

وصف البرنامج بصيغة مثال صريح

Write an OOP to add two areas of rectangle using (+) operator overloading.

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
class rectangle
{
private:
int length,width,area;
public:
void get()
{       cin>>length>>width;     }
rectangle()
{       area=0;     }
rectangle(int a)
{
    area=a;
 }
 rectangle operator + (rectangle r2)
 {
int a1 = length*width;
cout<< "area1 = "<<a1<<endl;

int a2= r2.length*r2.width;
cout << "area2 = "<<a2<<endl;

int a3=a1+a2;
return rectangle(a3);
}
void show()
{ cout << area; }
};
//////////////////////////////////////////////////////////////////////
void main()
{
rectangle r1,r2,r3;
r1.get();
r2.get();
r3=r1+r2;
cout << "area3 = "; r3.show();
}
```

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP | Class: Second

Week no.: 9 | Date: / /2024

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Binary Operators Overloading _ Comparison Binary Operator

b) Comparison Operators

Comparing operators consists of (>, <, >= ,<= ,==, !=).Examples bellow show some of these operator.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a comparison < Operator*** function using C++ language to compare between two ages

Example 3: وصف البرنامج بصيغة مثال صريح

Write an OOP to compare between two ages of person using (<) operator overloading.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class person
{
private:
int age;
public:
void get()
{
cout << "\nEnter age: ";
 cin >> age;
}

bool operator < (person p2)
{
if (age<p2.age)
return (true);
else
return(false);
}
};
//////////////////////////////////////////////////////////////
void main()
{
person p1,p2;
p1.get();
p2.get();

if( p1 < p2 ) //overloaded '<' operator
cout << "\n person1 is youngest than person2";
else
cout <<" \n person2 is youngest than person1";
cout << endl;
}
```

توقيع تدريسي المادة : توقيع مشرف المختبر:

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Inheritance is the process of creating new classes, called ***derived classes***, from existing or ***base classes***.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***an OOP to represent a Single Inheritance*** using C++ language for **counter** class. Also Learn how to build and implement ***an OOP to represent a Single Inheritance with overriding functions*** using C++ language for **employee** class

Example 1: وصف البرنامج بصيغة مثال صريح

Write an OOP to decrement the count variable in class counter using – operator and inheritance with ++ operator. The program includes a class called Counter and a class CountDn was derived from Counter class.

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class Counter      //base class
{
protected:
 int count;
public:
Counter()
{        count=5;    }

void operator ++ ()
{
    ++count;
}
void write()
{ cout<<count;}
};
////////////////////////////////////////////////////////////
class CountDn : public Counter     //derived class
{
public:
void operator -- ()
{
--count;
}
};
////////////////////////////////////////////////////////////
void main()
{
CountDn c1;
++c1;        ++c1;
cout << "\nc1=" ;        c1.write();

--c1;       --c1;       --c1;
cout << "\nc1=" ;       c1.write();
}
```

Example 2:

وصف البرنامج بصيغة مثال صريح

Write an OOP to read and write the information of employee using overriding functions .Create a class called employee which contains employee's name and number as private data items. Derived from class employee a class called manger which contains a salary of type float as private item

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class employee
{    private:
        char name[100];        long number;
public:
 void getdata()
{      cin >> name;        cin >> number;     }

void putdata()
{       cout << name;        cout << number;  }
};

class  manager : public  employee
{  private:
       float   salary;
public:
void  getdata()
{
     employee :: getdata();
        cin >> salary;
}
void putdata()
{
    employee :: putdata();
       cout << salary;
}
};
void  main()
{       manager m1;
         m1.getdata();
         m1.putdata();
}
```

توقيع تدريسي المادة :

توقيع مشرف المختبر :

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih Branch: NW and CS

Subject: OOP Class: second

Week no.: 11 Date: / /2024

Introduction:

مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Inheritance: Class Hierarchies

Objective:

تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a complete program to represent class hierarchies***

Example:

وصف البرنامج بصيغة مثال صريح

Example 1:-Write an OOP to model employ using inheritance *hierarchies*

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
class employee
{
private:
char name[30];     long number;
public:
void read()
{
 cout<<"\n enter name";     cin >> name;
 cout<<"\n enter number";    cin >> number;
}
void write()
{
cout << name;
cout << number;
}
};
//////////////////////////////////////////////////////////////////
class manager : public employee
{
private:
char title[30];

public:
void read()
{
employee::read();
cout<<"\n enter title"; cin >> title;
}
void write()
{
employee::write();
cout <<title;
}
};
```

```
//////////////////////////////////////////////////////////////////
class scientist : public employee
{
private:
int pubs;           //number of publications
public:
void read()
{
employee::read();
 cout<<"\n enter number of publications "; cin >> pubs;
}
void write()
{
employee::write();
cout <<pubs;
}
};
//////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////
class laborer : public employee
{

};
//////////////////////////////////////////////////////////////////
void main()
{
manager m;

scientist s;

laborer b;

cout << "\nEnter data for manager ";
m.read();

cout<<"\nEnter data for scientist ";
s.read();

cout << "\nEnter data for laborer ";
b.read();

cout << "\nData on manager ";
m.write();

cout << "\nData on scientist ";
s.write();

cout << "\nData on laborer ";
b.write();
}
```

توقيع تدريسي المادة : **توقيع مشرف المختبر:**

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: Nw and CS

Subject: OOP | Class: Second

Week no.: 12 | Date: / /2024

Introduction:

مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Inheritance: Multiple Inheritance*s*

Objective:

تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a complete program to represent multiple inheritances***

Example:

وصف البرنامج بصيغة مثال صريح

Write an OOP to model employee database using multiple inheritances.

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////
class student
{
private:
char school[30];     char degree[30];
public:
void getedu()
{
cout << " Enter name of school or university: ";     cin >> school;
cout << " Enter highest degree (Highschool, Bachelor's, Master's, PhD)earned \n";
cin >> degree;
}
void putedu()
{
cout << school;     cout << degree;
}
};
//////////////////////////////////////////////////////////////////
class employee
{
private:
char name[30];     long number;
public:
void getdata()
{
 cout<<"\n enter name";     cin >> name;
 cout<<"\n enter number";    cin >> number;
}
void putdata()
{
cout << name;
cout << number;
}
};
//////////////////////////////////////////////////////////////////
```

```
};
//////////////////////////////////////////////////////////////////
class manager : public employee, public student
{
private:
char title[30];

public:
void getdata()
{
employee::getdata();
cout<<"\n enter title"; cin >> title;
student::getedu();
}
void putdata()
{
employee::putdata();
cout <<title;
student::putedu();
}
};
//////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////
class scientist : public employee, public student //scientist
{
private:
int pubs; //number of publications
public:
void getdata()
{
employee::getdata();
cout << " Enter number of pubs: ";  cin >> pubs;
student::getedu();
}
void putdata()
{
employee::putdata();
cout << pubs;
student::putedu();
}   };
//////////////////////////////////////////////////////////////////
class laborer : public employee //laborer
{
};
//////////////////////////////////////////////////////////////////
void main()
{
manager  m;     scientist  s;       laborer  b;

cout << "\nEnter data for manager ";       m.getdata();
cout << "\nEnter data for scientist ";     s.getdata();
cout << "\nEnter data for laborer ";       b.getdata();

cout << "\nData on manager ";              m.putdata();
cout << "\nData on scientist ";            s.putdata();
cout << "\nData on laborer ";              b.putdata();
}
```

توقيع تدريسي المادة : توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih Branch: NW and CS

Subject: OOP Class: second

Week no.: 13 Date: / /2024

Introduction:

مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Virtual Function means ***existing*** *in appearance but* ***not*** *in* ***reality***.

Objective:

تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***an OOP to represent a virtual function*** using C++ language for **base and derived** classes.

Example 1:

وصف البرنامج بصيغة مثال صريح

Write an OOP to print "Base" , "Derv1" and Derv2 using virtual function

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
//////////////////////////////////////////////////////////////
class Base
{
public:
virtual void show()
{ cout << "Base\n"; }
};
//////////////////////////////////////////////////////////////
class Derv1 : public Base
{
public:
void show()
{ cout << "Derv1\n"; }
};
//////////////////////////////////////////////////////////////
class Derv2 : public Base
{
public:
void show()
{ cout << "Derv2\n"; }
};
//////////////////////////////////////////////////////////////
int main()
{
Derv1 dv1;
Derv2 dv2;
Base* ptr;
ptr = &dv1;
ptr->show();
ptr = &dv2;
ptr->show();
return 0;
}
```

توقيع تدريسي المادة: توقيع مشرف المختبر:

Introduction: مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Pure Virtual Function means ***existing*** *in appearance but* ***not*** *in* ***reality***.

Objective: تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***an OOP to represent a Pure virtual function*** using C++ language for **base and derived** classes

Example 2: وصف البرنامج بصيغة مثال صريح

Write an OOP to print "Base" , "Derv1" and Derv2 using pure virtual function

Lab Steps: كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////
class Base //base class
{
public:
virtual void show() = 0; //pure virtual function
};
//////////////////////////////////////////////////////////////////
class Derv1 : public Base
{
public:
void show()
{ cout << "Derv1\n"; }
};
//////////////////////////////////////////////////////////////////
class Derv2 : public Base
{
public:
void show()
{ cout << "Derv2\n"; }
};
//////////////////////////////////////////////////////////////////
int main()
{ Base* arr[2];

  Derv1 dv1;
  Derv2 dv2;
  arr[0] = &dv1;
  arr[1] = &dv2;
  arr[0]->show();
  arr[1]->show();
  return 0;
 }
```

توقيع تدريسي المادة :

توقيع مشرف المختبر :

Introduction:
مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Pure Virtual

Objective:
تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a complete program to check the person if successful or fail***.

Example:
وصف البرنامج بصيغة مثال صريح

Example 4: Write an OOP to read and check a person if successful or fail. Create a class called person which contains name of type string and derived a class student and a class professor from a class persons

Lab Steps:
كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>
//////////////////////////////////////////////////////////////////
class person //person class
{
 private:
char name[40];
public:
void getName()
{ cout << " Enter name: "; cin >> name; }
void putName()
{ cout << "Name is: " << name << endl; }
virtual void getData() = 0;      //pure virtual function
virtual bool is_success() = 0;   //pure virtual function
};
//////////////////////////////////////////////////////////////////
class student : public person
{
private:
float avg;
public:
void getData()
{
    person::getName();
cout << " Enter student's average: "; cin >> avg;
}
bool is_success()
{ if (avg >=50)
return true;
 else
 return false; }
};
//////////////////////////////////////////////////////////////////
```

```
class professor : public person
{
private:
int numPubs;
public:
void getData()
{
person::getName();
cout << " Enter number of professor's publications: "; cin >> numPubs;
}
bool is_success()
{ if (numPubs > 100)
return true;
 else
 return false; }
};
```

```
 void main()
 {

 person  *persPtr[5];
  int i;
  char ch;

 for (i=0;i<5;i++)
 {
     cout << "Enter student or professor (s/p): ";
          cin >> ch;

       if(ch=='s')
        persPtr[i] = new student;
      else
        persPtr[i] = new professor;

   persPtr[i]->getData();

   persPtr[i]->putName();

   if(persPtr[i]->is_success()==true )
     cout << " This person is successful\n";
   else
    cout << " This person is fail\n";
 }
 }
```

توقيع تدريسي المادة توقيع مشرف المختبر

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP | Class: Second

Week no.: 15 | Date: / /2024

Introduction:

مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Template Function

Objective:

تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a complete program to represent template Function***

Example:

وصف البرنامج بصيغة مثال صريح

Write an OO Program to find the absolute value using template function..

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>

template <class t>    //function template

void abs(t k)
{
if (k < 0)
cout<<-k;
else
cout<<k;
}
//-------------------------------------------------------------
void main()
{
int     a = 5;
int    b = -6;
long   x = 70000L;
long   y = -80000L;
double  n = 9.95;
double  m = -10.15;

 abs(a); cout<<endl;
 abs(b); cout<<endl;
 abs(x); cout<<endl;
 abs(y); cout<<endl;
 abs(n); cout<<endl;
 abs(m); cout<<endl;

}
```

توقيع تدريسي المادة: توقيع مشرف المختبر:

University of Technology/ Department of Computer Sciences

Lab(C++2)

Lecturer: Dr. Ekhlas Falih | Branch: NW and CS

Subject: OOP2 | Class: Second

Week no.: 15 | Date: / /2024

Introduction:

مقدمة لتعريف الجانب النظري المتعلق بالبرنامج

Template Class

Objective:

تعريف اهمية الجانب العملي للطالب بالنسبة للبرنامج

Learn how to build and implement ***a complete program to represent template Class***

Example:

وصف البرنامج بصيغة مثال صريح

Write an OOP that include a class called divide. Use the concept of template class to write the program. The divide class includes a and b as private, and two functions read() and a function div() to divide a on b. The main program includes the call of the objects of type integer and long.

Lab Steps:

كتابة الخطوات الاساسية للبرنامج كخوارزمية ان وجدت او بعض الايعازات المهمة

```
#include <iostream.h>

template <class  A>
class divide
{
private:
A a,b;
public:

void read()
{
    cin>>a>>b;
}
A div()
{
    return (a/b);
}
};
//////////////////////////////////////////////////////////////////////////
void main()
{
divide<int> d1;
d1.read();
cout<<d1.div();
cout<<endl;

divide<long> d2;
d2.read();
cout<<d2.div();
}
```

توقيع تدريسي المادة: توقيع مشرف المختبر: